DOCUMENT RESUME

ED 254 129 HE 018 025

AUTHOR	Balestri, Diane; And Others
TITLE	High Tech, Low Tech, No Tech: Three Case Studies of Computers in the Classroom.
INSTITUTION	American Association for Higher Education, Washington, D.C.
PUB DATE	Dec 84
NOTE	5p.
PUB TYPE	Viewpoints (120) -- Journal Articles (080)
JOURNAL CIT	AAHE Bulletin; p11-14 Dec 1984
EDRS PRICE	MF01/PC01 Plus Postage.
DESCRIPTORS	Case Studies; College English; *Computer Assisted Instruction; *Computer Oriented Programs; Computer Software; Economics Education; Higher Education; Medical Education; *Problem Solving; Programing; Technology Transfer; *Textbooks; *Writing (Composition)
IDENTIFIERS	*Spreadsheets

ABSTRACT

The use of computer technology in solving critical problems in education is described in three case studies. The "high tech" case is considered by Donald Thursh of the University of Illinois, who is creating a computerized textbook of pathology. The organization of an electronic text can be individualized to suit an instructor's emphasis, the student's knowledge level, or a clinician's specific needs. New information can be located and existing frame content can be easily revised. Extensive cross-referencing of information is accomplished without distracting users. The content expert works in ordinary English and maakes changes without programming competence. The "low tech" approach is described by Harold Cochrane of Colorado State University, who has adapted commercial spreadsheet software for use on microcomputers in his economics classroom. With these spreadsheets, students can solve problems and can see the intent of an economic model, its construction, and the sensitivity of the results to given information. Finally, the "no tech" case is considered by Diane Balestri of Bryn Mawr College, who has structured her freshman writing course around a metaphor of computer programming, with no machinery in the classroom. Students who are organizing and writing compositions are taught to use the same problem-solving and program-building techniques that they learn in a Pascal programming course. (SW)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

AAHE

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

ED254129

HE018025

HIGH TECH, LOW TECH, NO TECH
Three Case Studies of Computers in the Classroom

by Diane Balestri, Harold Cochrane, and Donald Thursh

U.S. DEPARTMENT OF EDUCATION
NATIONAL INSTITUTE OF EDUCATION
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

• Points of view or opinions stated in this document do not necessarily represent official NIE position or policy

Even as their deans and provosts learn to negotiate favorable vendor contracts, faculty must now devise uses for the hardware and software at their command.

As an incentive for innovative thinking about curriculum and pedagogy, the influx of computers onto American campuses could not be more timely. Recent proposals for curricular reform have lacked imagination or direction; students, moreover, have become alarmingly indifferent to every aspect of their education unrelated to a marketable GPA. Computer technology may help us find solutions to our teaching problems and promote change that is real improvement. But do computers effectively replace traditional tools of instruction? More importantly, how will the process of learning be changed when mediated by computer technology?

Rather than wait for the "Star Wars" institutions to develop their massive strategies and networks, we can seek preliminary answers to these questions through individual experiments in computer-based education already underway. The case studies described here have received funding from FIPSE and demonstrate how computer technology can be used in many different ways to solve critical problems in education.

Don Thursh is creating a computerized textbook of pathology: his is a high tech solution. Hal Cochrane has adapted commercial spreadsheet software for use on microcomputers in his economics classroom: his is a low tech solution. Diane Balestri has structured her freshman writing course around a metaphor of computer programming: with no machinery in her classroom at all, hers is a no tech solution.

The High Tech Case: Donald Thursh

Donald Thursh is professor of pathology at the University of Illinois College of Medicine at Urbana-Champaign.

I teach pathology, the science of disease. My students encounter learning problems typical of highly fact-intensive disciplines: the facts are numerous, they change rapidly, and they cannot be organized to suit every audience.

The traditional resource for the student of fact has been the scholarly textbook. However, even carefully prepared textbooks suffer inherent defects. First, organization is fixed; the author's sequencing of information may not seem logical to all readers, and thus may obstruct learning. Second, textbooks are outdated rapidly: they are costly and inconvenient to modify. Finally, even the most logically organized and current text, if structured for new learners, will not simultaneously be an adequate reference tool for practicing physicians.

The electronic textbook, or hypertext, such as the one my colleagues (especially Frank Mabry) and I have developed, offers solutions to these problems and provides other features not possible in print. In the electronic medium, a body of information, called a "frame," can be stored as easily as a computer file. Reordering the presentation sequence of frames is a trivial task for an experienced programmer; giving instructors, students, or professionals the option to reorder files themselves is a bit more complex, but still well within the state of the art.

This resequencing capability means that the organization of an electronic text can be individualized to suit an instructor's emphasis, a student's prior accumulation of knowledge, or a clinician's specific needs. Furthermore, new information can be immediately located in the appropriate frames, and existing frame content can be easily revised. Extensive cross-referencing of information on a computer is also easily accomplished, and, significantly, invisible. Thus, users are not distracted by the cross-referencing apparatus in the text unless they request more information. In the present version of our text, cross-referencing is author-directed, but we will soon give the machine the ability to accomplish this task itself.

Our electronic textbook contains a 500,000 word expanded outline of pathology. It has been operational on a PLATO system for about four years, where it has accumulated many hundreds of hours of student use. A new imple-

Students' enthusiasm for economics seems to be linked to their ability to connect the theoretical with the practical.

mentation is now operational under UNIX on a VAX-780 and a 68010-based microcomputer (CADMUS 9000 series). From the start, users have been able to ask questions, make comments on the information in context, and see the comments and questions of others. We have added other interactive features, such as the ability to question a student and judge the answer. Improvements under development include a simpler, more attractive, friendlier human interface and the ability to display graphics and videodisc-based illustrations in context.

One critical feature of our system has been the strict segregation of the computer program, which specifies how the computer is to manipulate information, from the information itself. Thus the content expert works in ordinary English and makes changes without prerequisite programming competence. Conversely, the knowledge management system can manage information unrelated to that for which it was initially developed. For instance, our system is beyond pathology and now manages knowledge-bases dealing with microcomputers and aids for the handicapped.

We are convinced that when electronic publishing is economically and academically viable, our system will enable us to offer not merely a computerized textbook of pathology but a fully cross-referenced mini-library of all biomedical science that will be able to communicate with conventional libraries and on-line bibliographic databases. Since this knowledge management system can be used in other subject areas, teachers in a variety of disciplines will have the opportunity to design computerized textbooks that they can modify to meet the needs of different audiences and to accommodate changes in information, purpose, emphasis, and student interest.

The Low Tech Case: Harold Cochrane

Harold Cochrane is associate professor of economics at Colorado State University.

I teach introductory economics and have observed that students have difficulty applying theoretical concepts to new situations. Perhaps the theoretical constructs are too abstract and remote from the students' range of experience.

Since students' enthusiasm for economics seems to be linked to their ability to connect the theoretical with the practical, economics can be more effectively taught by asking students to build this connection with their imaginations. "Imagine owning a small company." "Imagine you are retired and living on a fixed income." "Imagine you want to buy a house." In such role playing, students can set their own goals, use whatever economic signals they see as relevant, and make decisions accordingly.

In almost every instance, their own choices conform to the predictions of more formal models. Once they see this relationship, the theory is no longer abstract but is viewed as a shorthand representation of their own behavior.

I have been able to enhance this role-playing in my classroom dramatically by introducing microcomputers. Using commercial spreadsheet software, my students are able to set up and solve problems derived from their role-playing that are more realistic and extensive in scope.

The spreadsheets I use are familiar commercial programs. They resemble a blank tablet that can reflect any set of relationships the user desires. Columns of numbers can be added, subtracted, etc. Formulas can be stored, replicated, and edited. The results can be printed or graphed. A spreadsheet can be considered a high-level computer language that can be readily mastered by students with little or no previous exposure to programming. The key advantage is that students become active problem solvers.

Let me illustrate with an example from consumer theory. The life-cycle theory of household spending hypothesizes that individuals will attempt to plan purchases in a way that balances the current year's standard of living with that anticipated during retirement. Although this is a traditional treatment of the theory, even the brighter students tend merely to memorize equations drawn from textbooks that have already oversimplified the presentation.

With electronic spreadsheets, my students build life-cycle models. This does not require sophisticated training in mathematics; instead, students must think about the subject matter logically. When life-cycle theory is modeled on a spreadsheet, the intent of the model, its construction, and the sensitivity of the results to given information are visibly apparent to students. Furthermore, a spreadsheet model can be readily expanded to include more realistic assumptions. Students often add bits and pieces of their own: tax tables, stock portfolios, mortgage payments, etc.

Spreadsheets can also illustrate the complexities inherent in the

3

Good programming practice provides a concrete model for writing well-organized, idea-centered prose.

economy. For example, several models can be readily tied together to demonstrate various interdependencies, a particularly difficult concept for beginning economics classes. Spreadsheets can simulate complex interactions, reflecting the product of multiple decision makers responding to imperfect, conflicting signals.

Use of spreadsheets in the classroom is a fairly new phenomenon. Students are not simply recording data for the computer to manipulate; they are building their own models that the spreadsheet helps them develop and manipulate.

The students' models and the formal models of traditional textbooks begin to look alike when, in the process of building spreadsheets to solve a problem, students happen to discover an algorithm for the process. For example, in optimizing a household's temporal pattern of consumption, they discover trade-offs, wealth effects, and other elements of economic theory. Once they see that solutions can be more easily obtained using more elegant approaches, students are willing to pursue them. Thus, they begin to understand and use complex theoretical principles by generating manageable models of real experience on the computer.

The No Tech Case: Diane Balestri

Diane Balestri is associate dean and lecturer in English at Bryn Mawr College.

I teach composition, and there is no hardware in my classroom at all. Students are not even required to use wordprocessors, though I do not discourage them from doing so.

Learning about writing is being affected not by the mediation of computer tools, but by the mediation of a computing metaphor. I have developed a writing curriculum for freshmen who are concurrently learning to write computer programs in Pascal, a structured computer language. Students are taught to use the same problem solving and program-building techniques they learn in Pascal for organizing and writing compositions.

Good writing is a difficult skill to master, so why complicate it by interposing the learning of another difficult skill, programming?

Students generally like to write programs. The problem is that they do not like to write anything else. They are enthused by the problem solving aspect of programming, the ability to detect and correct errors, and the visible reward of a program that works.

For me as their teacher, however, structured programming is an appealing metaphor for composition because of its methodology. The strategies of structured programmers can be described as "techniques of construction:" programming is the process of building an integrated structure of unambiguous and logically related statements that accomplishes some activity for which it was designed. The strategy governing that process is called "top-down design;" programmers create the external structure of an entire program before developing the smaller pieces and coding them in the programming language. Through this metaphor, my students learn to focus attention on the design or construction of their arguments.

The idea that writing is a process of design and construction is a new one for my students. Their own idea is quite different. For them, papers are pages *filled up* with words, examples are *put in*. (How many freshmen each year ask their teachers, "Am I allowed to *put in* my own ideas?") From the programmers' perspective, this is writing "bottom up." As long as students liken writing to shovelling things into a hole, they are never going to find intellectual challenge or enjoyment in it. The programming metaphor with its implications of design and construction suggests a vigorous, more rewarding process; when you are done you have the builder's satisfaction of completing a workable structure.

This programming metaphor has many implications for teaching writing. Since the computer demands absolute perfection in syntax, spelling, and punctuation, for instance, students must appreciate the power of conventional usage. Less obvious, but more important, is students' early recognition that programs are *active* models for writing; like programs, essays should not merely record, but actually transform information in a purposeful way.

The most important implications of the programming metaphor for writing, however, are found in the concept of "top-down design," a term that encompasses the whole process of structured programming from problem identification to user-friendly commentary. Top-down design is founded on the creation of an algorithm, the concept or idea that governs the organization of a program. An algorithm-based approach to writing shapes the over-

We find our students mastering the traditional materials of our courses with greater ease and, perhaps most important, with greater enthusiasm and enjoyment.

all structure of the writing; it cannot be stuck in.

Once an algorithm is devised, it can be articulated as a scheme or structure. I have had particular success adapting Warnier-Orr diagrams, programmers' devices for organizing the elements of their programs, to the needs of my writing students. The program can then be coded in modular fashion; each module or unit is both internally integrated and logically connected to the problem as a whole. The resulting program is elegant; its argument is coherent, complete and concise.

Good programming practice provides a concrete model for writing well-organized, idea-centered prose. These are not the only qualities I value in good writing, but they are, I believe, the most important ones with which to begin.

* * *

Diverse as they are, our projects share one crucial feature: all attempt to affect the process of learning. For us, the computer is not a calculator but a processor of information that analyzes and organizes material into hierarchical and schematic structures. These structures can be invisible or graphically visualized, but they are always designed to be manipulated by the students themselves. The power of the computer's processing skill modifies and enhances students' learning processes as they practice learning, whether in the high-tech study carrel, in the low-tech classroom, or at the no-tech writing desk.

The result is that computer-assisted instruction, as we have conceived it, ultimately personalizes both learning styles and the body of knowledge thus acquired.

Computer technology has enabled our students to do more complex things more quickly. They have been given sophisticated, functional tools for controlling their own learning, organizing their knowledge, and expressing its coherence. Because these tools are also highly flexible and not domain-specific, students can, and ultimately must, manipulate them independently to master subject matter that they define for themselves. Despite our high expectations for student achievement, we find our students mastering the traditional materials of our courses with greater ease and, perhaps most important, with greater enthusiasm and enjoyment.

If computer technology has solved some old problems of instruction and learning, other problems—financial, technical, and philosophical—emerge. Computer projects like ours are not yet cost effective. High tech projects, especially, can easily exceed in scope the capacities of available hardware; dependence on sophisticated hardware can also create difficulties in dissemination of valuable projects to other institutions. Computer-equipped classrooms like Hal's can force expensive reductions in class size, in part because of equipment limitations but also because of the interactive nature of the class. For projects where the necessary technical support is lower, the costs are more often human, e.g. to buy faculty time for collaboration and personal development. Universities will have to solve these problems creatively, perhaps through extensive networking of both electronic and human resources.

There are also technical difficulties. The most significant of these involve management of the interfaces, the points at which students and faculty interact directly with computer hardware or software. Faculty, for instance, must be able to update Don's textbook in a form both easy for them to use and for the system to incorporate.

Orchestrating a friendly interface between the computer and the student user is particularly important to us because the computer systems built into our projects function as means to an end, and cannot be allowed to absorb students' attention. This problem becomes particularly acute in projects such as Diane's where students are obliged to be conversant with programming practices taught elsewhere.

Finally, we must ask whether efforts such as ours ultimately foster "machine-thinking," i.e. excessive insistence on mechanical correctness and "right" answers. Our pedagogy could encourage students to evade the ambiguous human and moral issues inherent in our subject matter, and to substitute games of abstract manipulation. Do we encourage analysis at the expense of judgment? The limitations of the computer in the realms of value and judgment must be acknowledged by the designer of any computer-based project. Learning cannot be conceived of as ending where we are suggesting it can productively begin, and institutions should not encourage faculty to narrow the focus of their teaching. The computer provides powerful tools for the acquisition and analysis of information: it remains for faculty to teach students how to use this information with subtlety and wisdom.

We will conclude optimistically, however; our projects have been invigorating to us and to our institutions as well as to our students. They have promoted interdisciplinary cooperation in their development at home and have created new and unexpected opportunities for interdisciplinary collaboration with colleagues across the country. These new internal and external networks, we expect, should be fostered by university administrators and by granting agencies as significant conduits for curricular renovation in the years ahead. ■